# Document made available under the Patent Cooperation Treaty (PCT)

International application number: PCT/AU05/000313

International filing date: 04 March 2005 (04.03.2005)

Document type: Certified copy of priority document

Document details: Country/Office: AU
Number: 2004901189
Filing date: 05 March 2004 (05.03.2004)

Date of receipt at the International Bureau: 22 March 2005 (22.03.2005)

Remark: Priority document submitted or transmitted to the International Bureau in compliance with Rule 17.1(a) or (b)

PCT/AU2005/000313

**Australian Government**

**Patent Office**
**Canberra**

I, LEANNE MYNOTT, MANAGER EXAMINATION SUPPORT AND SALES hereby certify that annexed is a true copy of the Provisional specification in connection with Application No. 2004901189 for a patent by VISION FIRE & SECURITY PTY LTD as filed on 05 March 2004.

COMMONWEALTH OF AUSTRALIA
AUSTRALIA
PATENT OFFICE

WITNESS my hand this
Sixteenth day of March 2005

LEANNE MYNOTT
MANAGER EXAMINATION SUPPORT
AND SALES

Regulation 3.2

**Vision Fire & Security Pty Ltd**

**AUSTRALIA**

**Patents Act 1990**

**PROVISIONAL SPECIFICATION**

for the invention entitled:

**"Testing of embedded systems"**

The invention is described in the following statement:

# TESTING OF EMBEDDED SYSTEMS

## FIELD OF INVENTION

The present invention relates to the field of embedded or distributed systems. In one form, the present invention relates to a method and apparatus for the testing of embedded devices. It will be convenient to hereinafter describe the invention in relation to the use of a method and apparatus employing, in part, CASE (Computer Aided Software Engineering) tools to model the behaviour of an embedded device or system referred to herein as a Device Under Test (DUT), however it should be appreciated that the present invention is not limited to that use only.

## BACKGROUND ART

Throughout this specification the use of the word "inventor" in singular form may be taken as reference to one (singular) or all (plural) inventors of the present invention.

The inventor has identified and considered the following technologies.

In general, an embedded system is a system within a larger system. Embedded systems may be implemented on a single integrated circuit or as scaled-down software code. An embedded system typically has a specialised function with programs stored on ROM. Examples of embedded systems are chips that monitor automobile functions, comprising engine controls, antilock brakes, air bags, active suspension systems; industrial process devices, for example comprising robotic systems in production engineering applications or medical applications; environmental systems, comprising pollution monitoring devices etc; entertainment systems and; property protection systems, comprising for example security devices, fire prevention or smoke detection devices and combinations thereof. Ordinarily, everything needed for those functions is custom designed into specific chips and, there may be no external operating system required. Many embedded systems are produced in the range of tens of thousands to millions of units. Reducing costs in all aspects of design to actual production, installation and maintenance is therefore a major concern. There may

be many different CPU architectures used in embedded designs. This contrasts with the desktop computer market, which is limited to only a small number of competing architectures, such as, Intel™'s x86, and the Apple™/Motorola™/IBM™ PowerPC™. A common configuration for embedded systems is the "system on a chip", an application-specific integrated circuit. It is also notable that user interfaces for embedded systems vary to a large extent. The resultant complexity of most embedded systems requires that generally they are developed in teams by several developers at the same time using different computer platforms. Furthermore, high performance micro-controllers which are now available may provide developers with highly complex software solutions for embedded system design. Managing the complexity of software components and their inter-workings has therefore also become a major concern.

Rational™ Robot is a PC application designed for the design and execution of tests on PC applications. Therefore, in an attempt to provide a test environment for embedded systems by use of Rational™ Robot, a Data Driven Engine (DDE) is required to deal with the complexity of generating tests within a PC environment and translating those tests into a syntax that may be interpreted by additional interface hardware.

Visio™ is an application where a message sequence chart (msc) representing a test case or sequence may be constructed and then via interfacing Visual Basic code representing the Visio™ msc to Labview™, a graphical programming language tool. The Labview™ functions may then interpret this code and execute the test on a DUT. It is to be noted that Labview™ being the programming environment, still needs to interface physically and electrically to the DUT via specific I/O hardware. There is also an issue of having to reconstruct from scratch the entire msc every time there is a change to the underlying test sequence requirements.

A product from Lucent™ known as UBET employs an enhancement of msc's, namely, HMSC (Hierarchical Message Sequence Charts). The limitations of the Visual Basic scripts are removed in this implementation, however, the inventor has identified that there still remains the issue of reconstructing from

scratch the MSC's and HMSC's with every iteration in requirements or change in requirements of a test sequence for embedded systems.

Generally, "black box" test environments do not exist for embedded systems, in particular, systems that are not built on or do not employ industry standard communications protocols. In a niche market, such as is filled by the fire and smoke alarm products developed by Vision Fire & Security Pty Ltd and marketed under trade marks such as VESDA®, the communications protocols are developed in house and are not serviced by major industry developers of automation systems, for example, such as TTCN2 and TTCN3. Thus no means exists to leverage off the activity of cooperating industries and user groups.

In general, test automation environments such as Rational™ Robot are contemporaneous with other automation environments such as WinRunner™ and other PC application and GUI focused applications. The inventor has identified that these environments are designed in a manner, which may not communicate with, stimulate or control embedded systems. The inventor has further identified that attempts to retrospectively adapt these environments through the addition of interpreters or hardware interfaces necessitates very large efforts in terms of scripting and definition of key words and phrases. Moreover, the inventor has identified that test designing within these environments requires repetitive manual labour. Overall, the inventor has identified use of these environments to be time consuming, difficult to maintain and labour intensive in generation and updating and, unable to adequately deal with any changes in requirements and project scope.

The inventor has further identified that, under such environments, bridging the divide from formally defined tests to physically interface to a product that is controlled via embedded software and automatically execute the tests introduces difficulty given that very few user accessible interfaces are generally available for stimulus and reading of responses from embedded DUTs.

Any discussion of documents, devices, acts or knowledge in this specification is included to explain the context of the invention. It should not be taken as an admission that any of the material forms a part of the prior art base or

the common general knowledge in the relevant art in Australia or elsewhere on or before the priority date of the disclosure herein.

An object of the present invention is to automatically execute functional tests for embedded systems at a system-wide or black box level in a manner that is responsive to changing test requirements.

A further object of the present invention is to alleviate at least one disadvantage associated with the prior art.

The scope of applicability of the present invention will become apparent from the detailed description given hereinafter. However, it should be understood that the detailed description and specific examples, while indicating preferred embodiments of the invention, are given by way of illustration only, since various changes and modifications within the spirit and scope of the invention will become apparent to those skilled in the art from this detailed description.

**BRIEF DESCRIPTION OF THE DRAWINGS**

Further disclosure, objects, advantages and aspects of the present invention may be better understood by those skilled in the relevant art by reference to the following description of preferred embodiments taken in conjunction with the accompanying drawings, which are given by way of illustration only, and thus are not limiting to the scope of the present invention, and in which:

Figure 1 is a block diagram schematically illustrating an automated model architecture and test environment in accordance with an embodiment of the present invention;

Figure 2 is a block diagram schematically illustrating architecture of a generic test execution engine in accordance with an embodiment of the present invention;

Figure 3a illustrates a file showing the results of a comparison between modelled device behaviour and actual device behaviour wherein a device has passed an automated test in accordance with an embodiment of the present invention;

Figure 3b illustrates a file showing the results of a comparison between modelled device behaviour and actual device behaviour wherein a device has

failed an automated test in accordance with an embodiment of the present invention; and

Figure 3c illustrates a file showing the results of a masked comparison between modelled device behaviour and actual device behaviour wherein a device has passed an automated test in accordance with an embodiment of the present invention.

**DETAILED DESCRIPTION**

By way of background, a number of terms are here defined.

DUT Device Under Test. A target embedded device or, an embedded system comprising one or more embedded devices that is the subject of tests. Furthermore, a DUT may comprise one or more embedded systems being the subject of tests.

Model An approximation of selected aspects of the behavioural characteristics of a real-world embedded DUT and real world influences on the embedded DUT. A model may use Architectural Level Descriptions (ALD) or Definitions, State Transition Definitions, Extended Message Sequence Charts (EMSC) and Data Type Definitions to provide an abstract representation of the physical interface, real world environmental influences and constraints upon an embedded DUT.

Output Port An output port is used in a model to represent a physical output on the DUT being modelled. The physical outputs of a DUT may comprise LED's, relay contacts, Open Collector, Wet or Dry Outputs, Communications Ports, for example, RS-232 employing ADCP etc, and Ethernet Ports.

TDF Test Data File. A formatted ascii text file containing Test Vectors generated based on definitions contained in a test configuration file (TCF). There may be two versions of a TDF utilised in embodiments of the present invention, an input test vector only TDF and a input/output test vector (response) TDF created when the input only test vector TDF is passed through the model.

Test Vector An ordered set of input, output, or input and output values defining input and/or output information. For example a test vector may comprise a line within a TDF representing a command set of inputs or input/outputs.

Test Vector Pair A Test Vector pair is the same as a test vector, however, the outputs represented are valid at that point in time indicated by the value of a predefined timing reference provided in accordance with embodiments of the present invention. The predefined timing reference may be provided by way of a Timing Port.

Clock Pulse A clock pulse is an arbitrary but fixed timing period employed by a device model to synchronise activity within the model resulting in propagation of variables and transition firing. The period of time within the model during which inputs are sensed, calculations computed and outputs generated.

Config Port A port that is used by a test engineer/test designer to identify for the automated execution engine of an embodiment of the present invention that the parameter associated with the port is a configuration parameter and should be applied before any stimulus is applied to the input ports.

GEE A Generic Execution Engine in accordance with an embodiment of the present invention.

Execution Port A port that is used by the GEE to initiate and maintain communication with a DUT Specific Interface.

TCF Test configuration file. A set of instructions that define the configuration parameters, stimuli parameters (patterns), the permutation and combinations of all input ports with port input patterns, test repetition rates and test group repetition rates, and the definition of the explicit test cases under the preceding settings which results in the generation of input (stimuli) test vectors associated with input only test data sets, which may be in the form of test data files (TDF's). The TCF therefore effectively comprises the test sequence rules. The Autofocus™ modelling tool provides for the creation of GCF's (Group Configuration Files), which may be viewed as a test sequence configuration facility. However, the TIG (Test Input Generator), used to create GCF's in Autofocus™, is unable to produce configuration files for floating models using floating point arithmetic. The TCF's of the present invention provide for what is known in graph theory as a "guided tour". That is the State Transition Diagram of a model is considered as a graph through which the execution of test sequences is

guided in such a manner as to directly influence and control the mode and manner in which testing progresses. As a result, the present invention may be in control of what gets tested, how it gets tested and to what degree (coverage) it is tested. This features is not a natural consequence of the GCF from an Autofocus™ TIG, where a GCF will result in a random unguided mode of execution.

Input Port An input port is used in a DUT, and in particular a DUT model to represent a physical input on the DUT being modelled.

Timing Port A port that is used by the test designer to indicate to the automated execution engine at what time, in accordance with a predefined timing reference, to sample the outputs of the DUT (modelled or actual) for matches of stabilised output vectors with corresponding input vectors. Furthermore, in the case of obtaining outputs from an actual DUT when provided with the input vector (which has a corresponding model generated output vector) the DUT output is sampled for the output vector or response, which is then compared against the model generated output vector of the same input vector. The DUT response may therefore be determined for its equivalence to that of the model for the same stimuli. The Timing Port is a port that doesn't necessarily exist on the DUT. The execution engine executes a wait period determined by the predefined timing reference indicated on the timing port directly after applying stimulus to the input port.

In one aspect, a preferred embodiment of the present invention provides in an embedded DUT testing system comprising apparatus adapted to compare actual DUT input/output vector pairs with modelled DUT input/output vector pairs, a logical connection port adapted to indicate a predefined timing reference for determining a point in time at which to sample an output vector as the corresponding output vector in an input/output vector pair. The sampled output vector may be a modelled or actual output vector of one or more of:

a smoke detector;

a fire detector;

a security device;

a medical device;

a biological tissue processing device;

an industrial process device.

In another aspect, a preferred embodiment of the present invention provides a method of testing at least one embedded DUT comprising the steps of:

a) determining a test configuration parameter set comprising predefined DUT test sequence rules;

b) determining a first data set comprising input test vectors based on the test configuration parameter set;

c) processing the first data set in a DUT model to determine output test vectors wherein the output test vectors comprise DUT model generated responses to the input test vectors;

d) processing the first data set and the output test vectors to determine a second data set comprising pairs of stabilised input and output test vectors;

e) communicating the stabilised input test vectors to at least one DUT via a DUT independent interface so that the at least one DUT is stimulated by the stabilised input test vectors to produce DUT output vectors;

f) determining a third data set comprising the stabilised input vectors and corresponding DUT output vectors;

g) comparing the third data set with the second data set to determine a comparison of actual behaviour to modelled behaviour of the at least one DUT.

Step d), above, may further comprise the steps of:

h) parsing the output test vectors with the first data set in accordance with a predefined timing reference in which the predefined timing reference determines a point in time to sample an output test vector as a stabilised output test vector;

i) matching each stabilised output test vector to a corresponding input test vector to form pairs of stabilised input and output test vectors.

The predefined timing reference may be derived from a logical connection port as described above. Further, the predefined timing reference may comprise one of:

one delay period common to all input test vectors, and;

a predetermined delay period for each input test vector.

The DUT independent interface may comprise an interprocess communication protocol utilising one of:

TCP/IP, and;

Active-X.

The DUT model may comprise one or more of the following abstraction techniques:

architectural level descriptions;

data type definitions;

state transition diagrams;

extended Message Sequence Charts.

The data sets described above may comprise test data formatted files.

The test configuration parameter set described above may comprise a test parameter configuration file.

The DUT may comprise one or more of:

a smoke or fire detector;

a security device;

a medical device;

a biological tissue processing device;

an industrial process device.

In yet another aspect, a preferred embodiment of the present invention provides apparatus adapted to test at least one embedded device (DUT), said apparatus comprising:

processor means adapted to operate in accordance with a predetermined instruction set,

said apparatus, in conjunction with said instruction set, being adapted to perform the method steps as described herein.

In still another aspect, a preferred embodiment of the present invention provides a generic test execution engine for testing at least one embedded DUT comprising:

a test data interface comprising parsing means for parsing modelled output test vectors with predefined input test vectors in accordance with a predefined

timing reference to determine a test data file comprising pairs of corresponding stabilised input and output test vectors;

a DUT interface comprising addressing means for addressing, via data ports, test vectors to respective DUT's in accordance with an address identifier within the test data file, and communication means for receiving DUT output vectors in response to the test vectors;

a test result interface comprising processing means for processing respective pairs of stabilised input and output test vectors and corresponding received DUT output vectors. As noted above, the predefined timing reference may be derived from a logical connection port as described herein. The predefined timing reference may determine a point in time to sample an output test vector in order to be determined as a stabilised output test vector for a corresponding input test vector.

The generic test execution engine as described herein may be utilised for testing at least one DUT wherein the at least one DUT comprises one or more of:

a smoke or fire detector.

a security device;

a medical device;

a biological tissue processing device;

an industrial process device.

In a preferred embodiment of the present invention an embedded device testing system is provided comprising:

a generic test execution engine as described herein; and,

logging means operatively associated with the test result interface for recording an indication of a comparison between respective pairs of stabilised input and output test vectors and corresponding received DUT output vectors. The recorded indication may comprise a formatted file comprising the following fields:

DateTimeStamp;

Evaluated Result;

A Binary String representing a logical comparison of each port value between respective test and DUT output vectors;

Port Identifier;

Port value comprising one of matched port value and expected value/actual value.

In still a further aspect, a preferred embodiment of the present invention provides a data format for use in an embedded DUT testing system, the data format comprising an input test vector field correspondingly paired with an output test vector field wherein:

the input test vector field comprises predefined input information;

the output test vector field comprises stabilised output information determined by sampling output information, provided in response to corresponding predefined input information, at a point in time determined by a predefined timing reference. The predefined timing reference may be derived from the predefined input information. Further, the predefined input information may comprise command set input data and the predefined output information may comprise command set output data.

In yet a further aspect, a preferred embodiment of the present invention provides an embedded DUT testing system data file comprising a data format as herein described.

In still another aspect, a preferred embodiment of the present invention provides a data transmission packet format for use in an embedded DUT testing system comprising a data format as herein described.

In yet another aspect, a preferred embodiment of the present invention provides a computer program product including:

a computer useable medium having computer readable program code and computer readable system code embodied on said medium for testing at least one embedded DUT within a data processing system, said computer program product comprising:

computer readable code within said computer useable medium for performing the method steps of a method as herein described.

In essence, the present invention stems from the realisation that determining stabilised output data by, for example, providing a predefined timing reference to determine a point in time in which to sample or accept a given output, of either a modelled DUT or an actual DUT in response to a given input, allows for a realistic evaluation of the behaviour (modelled or actual) of a DUT, which is constrained to operate in a real environment. Determining realistic or stabilised output allows reliable test data to be determined much sooner than in the case of an ad hoc approach to determining the output data corresponding to a given input. The stabilised output data may then be combined with its corresponding input data to provide a data format with which to interface, drive and subsequently test a plurality of DUT's. The plurality of DUT's may be multiples of the same DUT or alternatively, multiple DUT's being different devices whether they provide the same function or not. In a preferred embodiment of the invention, one or a plurality of the same DUT, eg a fire detection device, is/are subjected to the same or different tests.

The present invention has been found to result in a number of advantages, such as:

- Rapid prototyping of the functionality and behaviour specified for new products
- Dynamic interaction with the DUT model results in early detection of requirement defects
- Based on supplied user (test engineer) defined or handcrafted configuration files containing test sequence rules, an almost unlimited number of tests and results can be automatically generated based on the DUT model behaviour
- No reliance on or need for Data Driven Engines or similar as all necessary variable names are maintained from within the model and linked to Labview™ VI's.
- Automatic execution of the resultant executing test data sets, which may be in the form of Execution Test Data Files (ETDF's) via a graphical

programming language interface to one or more I/O devices and subsequent reporting is automatic

- Parsing and checking for differences between model predictions and actual results is automatic
- Increased requirement/feature interaction (test sequence/rules) coverage
- Increased behavioural tests
- Speed of test execution and reporting
- Elimination of human interaction during automated test execution
- Improved repeatability in, for example, regression test execution
- Facilitation of complete regression test execution of all generated tests at minimal expense

The inventor considers one major benefit or advantage of the invention is the automation of all manual, time intensive, tedious and error prone test generation and execution tasks.

With reference to Fig 1, an automated modelling and test environment 10 is now generally described as it relates to a preferred embodiment of the present invention. Dynamic, executable and configurable abstracted models 101 are created from which are derived responses corresponding to input stimuli 102. The models are predicated on specific test sequence rules 100 in the form of test configuration parameter sets, which may be embodied in the form of a test configuration file TCF as defined above. Generally speaking, the appropriate pairing by means of parsing 103 the input stimuli to stabilised output response of a model in accordance with an embodiment of the present invention provides, in a sense, a test oracle 108 for reference against actual DUT 105 behaviour. A test oracle 108 as such, in the context of this specification, provides a means by which the dynamic behaviour of an embedded device/system under test DUT 105 may be dynamically interacted/interfaced 106 with and interrogated and from which a priori knowledge of potential responses to input stimuli may be derived. The resultant sets or files of input to stabilised output may be retained in a "golden log" or oracle 108 as a prediction against which the actual responses 107 of the DUT

105 for the same stimuli 104 may be evaluated by a comparison 109 and from which a deviation from expected behaviour based on the original test sequence rules/requirements set of the actually implemented devices may be observed. From a separate file that only has input or stimuli vectors a drive TDF file may be created 104 and used to drive an interface environment 106 or harness that provides the electrical and control interfaces to the device(s)/system(s) under test, DUT 105. By means of capturing the actual device under test responses to the input stimuli derived from the model as described above, an output file or log 107 of stimuli to response may be generated. This actual output file may then be compared 109 to the "golden file" of input to output derived from the model, namely, the test oracle 108. Suitable scripts, for example, written in Perl or Labview™ elements may implement the comparison. The differences identified by this comparison may be noted and investigated for defects within: the implemented DUT 105; defects in the requirements 100; or, defects in the test environment generally.

The testing environment 10 of embodiments of the present invention rather than requiring manually hand crafting test sequences based on individual requirements in a very static sense which then need to be associated with a unique data driven engine or similar and automation of test through a GUI based engine testing design is instead based on dynamic behaviour expressed in a model 101 that may be interacted with. Hence test design may be a by-product of interacting with a model 101 that encapsulates the essence of the behaviour of the device or product in question DUT 105. By providing a model 101 with a series of inputs, the model produces outputs that are predictions of the actual behaviour of the actual DUT 105. Thus by appropriate application of input vectors in accordance with the present invention and generating tests, small to extremely large sets of tests can be quickly generated. In a preferred embodiment of the present invention, it has been identified that combinatorial generation of tests per se may be provided by commercial products such as Autofocus™. The present invention limits, in a crude but effective way, the domain space explosion inherent in combinatorial generation through Autofocus™ by explicitly crafting the

parameters (within a TCF) that represent the input domain carefully so as to minimise this domain explosion. It is a crude approach in as much as it may be reliant on the knowledge of the test designer. Other solutions to the domain state explosion caused by combinatorial effects have been postulated, this includes application of CLP (Constraint Logic Programming) techniques, model proving tools such as SATO etc. However all these solutions so far reside very heavily in the research domain and have not yet made an effectual transition to industrial practice. Maintaining a record of tests is resolved by recognising that a model 101 may be used as the repository of the behaviour that gave rise to the tests generated. By altering the model 101 to keep pace with changing requirements 100 and reusing the input stimuli, all tests may be updated as required with almost no human interaction. In a straightforward approach, the model is an abstraction that is moulded to represent the DUT behaviour in question. The GEE and DSI's (Device Specific Interfaces) have knowledge of the terms and parameters and ports employed by the model. As long as the model does not change the implementation of these terms, parameters, ports etc. then there may be no issue. If changes are made in the model by the amendment of STD's, SSD's etc then the GEE and DSI's need to be amended to account for this change. This may be implemented, for example, in Labview™. By execution of tests 104 through, for example a Labview™ environment, with automatically based sets of stimuli and capturing the output results for automatic comparison with predicted output again eliminates or reduces dramatically human interaction.

Given that models are re-useable then the reuse of model components is conducive to speeding up development of new models for new products DUT's that may be of the same technology family. Models are reusable in the sense that if the behaviour exhibited by the model or a component of the model is consistent with the behaviour intended by a new or different implementation of a DUT for which the original model was designed, then it is possible to re-apply elements of the model directly or after some minor amendment. As noted above, in accordance with a preferred embodiment, state based behavioural models are created in a modelling environment 101 such as Autofocus™. A handcrafted or

user defined 100 Test Configuration File (TCF) is parsed through the model preferably in a compiled C version which is an executable giving rise to a Test Data Format (TDF) file which represents the input test vectors generated based on the information contained in the TCF. This TDF of input only stimuli (test vectors) which comprises the same input set of stimuli as the golden set is generated and parsed, time stamped, filtered and formatted and the resultant TDF from this process is utilised to drive 104 an automation test interface 106 developed, preferably in a Labview™ environment. Alternatively other modules could provide the same functionality that Labview™ provides, namely a programming environment per se, for example, an ADA test environment. Labview™ conveniently has associated with its programming environment, off the shelf I/O solutions that comprise drivers that facilitate direct integration with the Labview™ applications. It is, however, possible to develop a similar solution in ADA, C, Perl etc. A separate TDF is generated, which comprises input vectors (stimuli) to output vectors (responses) generated by the model based on the behaviour represented by the model. The resultant TDF described here is retained as the golden set of test input 108 to test data against which the output 107 of the DUT 105 will be compared. In a preferred embodiment Labview™ Virtual Instruments (VI's) are developed 106 that are DUT 105 specific retaining knowledge of the unique port names and variables utilised in the model and have knowledge of the specific command language of the DUT 105. Physical interfacing to the DUT 105 may be via specific I/O cards. The VI's may be controlled and coordinated by a master VI or test "manager" 106. The master VI has the task of controlling the timing and execution of the tests as represented in the parsed and filtered TDF, or drive TDF (DTDF) mentioned earlier. By reading the input vectors represented in the DTDF the VI's generate the appropriate electrical stimuli necessary to stimulate the inputs of the device DUT 105 and to configure it. The outputs of the DUT 105 are subsequently monitored and the responses recorded 107. A log or execution TDF (ETDF) in a format consistent with the DTDF is consequently generated through the course of the test execution. Post test execution the DTDF and ETDF may be automatically compared 109 via an appropriate script or in a

Labview™ environment. The differences in response from the actual DUT output as compared to that of the original TDF from the compiled model or oracle 108 are flagged as potential defects or deviations from expectations based solely on the original product function and performance requirements.

**Parsing of a TDF File**

Below is an example of an output TDF file after an input only TDF has been passed through a model. The preferred format of the TDF below is

<input port >?<pattern>; ... <output port>!<pattern>;

in this example Ports 1-3 are input ports and Ports 4-6 are output ports.

```
Port1?x;Port2?y;Port3?z;Port4!;Port5!;Port6!;
...
Port1?x;Port2?y;Port3?z;Port4!a;Port5!;Port6!;
...
Port1?x;Port2?y;Port3?z;Port4!a;Port5!b;Port6!;
...
```

repeated n number of times as the input vectors are passed through the model

```
...
Port1?x;Port2?y;Port3?z;Port4!a;Port5!b;Port6!;
...
```

until eventually all of the output ports (represented by <portname>!<value>) are populated.

```
...
Port1?x;Port2?y;Port3?z;Port4!a;Port5!b;Port6!c;
Port1?x;Port2?y;Port3?z;Port4!d;Port5!b;Port6!c;
Port1?x;Port2?y;Port3?z;Port4!a;Port5!b;Port6!c;
...
```

however, even at the point of populating all output ports, the values contained therein may change in the time it takes to clock through the entire model. This may vary dependant upon how the model was implemented, that is, variation may be dependent on the nature of the device being modelled. For example, a smoke detector will not immediately produce a true output value at the time a stimulus is provided at its input. However there will come a time when the output stops changing becoming a stabilised output and it is this vector that must be used to model and test the device.

```
...
Port1?x;Port2?y;Port3?z;Port4!a;Port5!b;Port6!c;
Port1?x;Port2?y;Port3?z;Port4!a;Port5!b;Port6!c;
Port1?x;Port2?y;Port3?z;Port4!a;Port5!b;Port6!c;
Port1?x;Port2?y;Port3?z;Port4!a;Port5!b;Port6!c;
Port1?x;Port2?y;Port3?z;Port4!a;Port5!b;Port6!c;
Port1?x;Port2?y;Port3?z;Port4!a;Port5!b;Port6!c;
```

Eventually a point in time arrives when the outputs have stabilised and a matching input and output vector (a vector pair) may be attained. The issue is at what point in time after the input stimulus is applied will the output be valid? A number of different methods have been employed by the inventor as follows:

1. Wait a fixed period of time – this may prove to be quite inefficient and relies on the fact that each test vector pair doesn't rely on what state the last vector pair left the system in. In a working system this was flawed because it didn't allow the model designer – the true test engineer – to create sequence's (inter linked vector pairs)

2. Wait until the actual outputs satisfy a predefined match of the vector pair. This is also flawed because it requires waiting until the output matches. However, it is usual to employ some kind of time out in this situation otherwise if the outputs never match, there will be no progressing through the remainder of the vector pairs. Employing a timeout has the same symptoms as those in 1 above.

3. Count the number of lines within a TDF before the Test Vector pair and use that to determine the time before the outputs are valid. This is may not be valid since the number of test vectors will vary based upon the implementation of the model. Two models of the same DUT may be implemented in different ways that will require a different number of clock pulses to generate the same test vector pair.

4. Have the test vector indicate a predefined period of time before it's output vector is valid. This is in essence achieved by creating extra timing ports, for example, in the model. The timing ports indicate to the automated execution engine what period of time to wait before sampling the outputs of the system under test. This underlies a preferred embodiment of the present invention.

**Generic Automated Test Execution**

It is desirable to implement an execution engine in a generic fashion that allows the re-use of the same architecture across multiple products and projects, DUT's. This requires a test execution engine that doesn't have any knowledge of the DUT or the interface executing the test.

Executing automated tests involves a number of steps,

Configure DUT

Start test on DUT

Implement a system "Wait" for a period time until outputs can be verified

Measure output values

Evaluate results against the expected values.

**Generic Execution Engine Architecture**

The architecture of the Generic Execution Engine (GEE) 20 in accordance with a preferred embodiment is shown in Fig 2. This architecture 20 supports either multiples of the same DUT 23 to facilitate parallel test execution or different DUT's 23 being tested simultaneously.

A number of advantages are provided by the GEE 20 firstly, it allows for accelerated life tests, greater test coverage in the same period of time, independent testing of multiple DUT's or multiple instances of the same DUT. A further advantage is in terms of reliability, ie should one DUT fail it is independent of all other DUT's and tests being executed on them.

The architecture for the GEE 20 details a number of interfaces

(i) TDF-GEE Interface

(ii) GEE-DUT Interface

(iii) GEE-Result Log Interface

**TDF-GEE Interface (i)**

The TDF-GEE interface resides between the GEE 20 and model generated TDF files 21. This interface provides the function detailed above in relation to the parsing of a TDF file.

**GEE-DUT Interface (ii)**

This is a communication interface utilising TCP/IP in a preferred embodiment. However, any other interprocess communication method may be employed such as, for example, Active-X comprising for example, DCOM or COM. The socket address for a given DUT 23 is identified by a TCP/IP_Address port, which is preferably contained in a TDF 21.

The GEE 20 manages the GEE-DUT interface and provides the following services to DUT specific Interface(s) 22,

(i) Test Vector Pair – provide the test vector pair to the DUT specific Interface 22

(ii) Restart Test – Re-send test vectors starting from the first test vector pair in the TDF 21

(iii) Add result to log 24 – DUT specific Interface 22 provides test vector pair and resultant vector pair for processing by GEE to provide formatted actual data to results log 24

**GEE-Result Log Interface (iii)**

The format of the Result Log 24 is an ascii format file. The file name used may be generated by concatenating together the value of the DUT 23 port in the TDF 21 and a time and date stamp.

The file contains data formatted in the following way,

- DateTimeStamp,
- Evaluated result +++ = PASS,
  ++- = PASS (masked port failure encountered)
  --- = FAIL,
  *** = Error encountered during test
- Binary String representing a logical comparison of each port value between the Test Vector Pair and Resultant Vector Pair. The user is able to mask the comparison of any ports that they choose,

- Each port is listed and if the data matches only one value is shown. Otherwise, if values don't match then the expected value is presented first followed by the actual value.

Examples of each sequence Results Log file described above are shown in figures 3a to 3c, respectively.

While this invention has been described in connection with specific embodiments thereof, it will be understood that it is capable of further modification(s). This application is intended to cover any variations uses or adaptations of the invention following in general, the principles of the invention and including such departures from the present disclosure as come within known or customary practice within the art to which the invention pertains and as may be applied to the essential features hereinbefore set forth.

As the present invention may be embodied in several forms without departing from the spirit of the essential characteristics of the invention, it should be understood that the above described embodiments are not to limit the present invention unless otherwise specified, but rather should be construed broadly within the spirit and scope of the invention as described. Various modifications and equivalent arrangements are intended to be included within the spirit and scope of the invention. Therefore, the specific embodiments are to be understood to be illustrative of the many ways in which the principles of the present invention may be practiced. In statements of the invention herein, means-plus-function clauses are intended to cover structures as performing the defined function and not only structural equivalents, but also equivalent structures. For example, although a nail and a screw may not be structural equivalents in that a nail employs a cylindrical surface to secure wooden parts together, whereas a screw employs a helical surface to secure wooden parts together, in the environment of fastening wooden parts, a nail and a screw are equivalent structures.

"Comprises/comprising" when used in this specification is taken to specify the presence of stated features, integers, steps or components but does not preclude the presence or addition of one or more other features, integers, steps, components or groups thereof."

## ABSTRACT

The present invention relates to the field of embedded or distributed systems. In one aspect, the present invention provides in an embedded DUT testing system comprising apparatus adapted to compare actual DUT input/output vector pairs with modelled DUT input/output vector pairs, a logical connection port adapted to indicate a predefined timing reference for determining a point in time at which to sample an output vector as the corresponding output vector in an input/output vector pair.

In another aspect the present invention provides for the testing of at least one embedded DUT comprising:

a) determining a test configuration parameter set comprising predefined DUT test sequence rules;

b) determining a first data set comprising input test vectors based on the test configuration parameter set;

c) processing the first data set in a DUT model to determine output test vectors wherein the output test vectors comprise DUT model generated responses to the input test vectors;

d) processing the first data set and the output test vectors to determine a second data set comprising pairs of stabilised input and output test vectors;

e) communicating the stabilised input test vectors to at least one DUT via a DUT independent interface so that the at least one DUT is stimulated by the stabilised input test vectors to produce DUT output vectors;

f) determining a third data set comprising the stabilised input vectors and corresponding DUT output vectors;

g) comparing the third data set with the second data set to determine a comparison of actual behaviour to modelled behaviour of the at least one DUT.

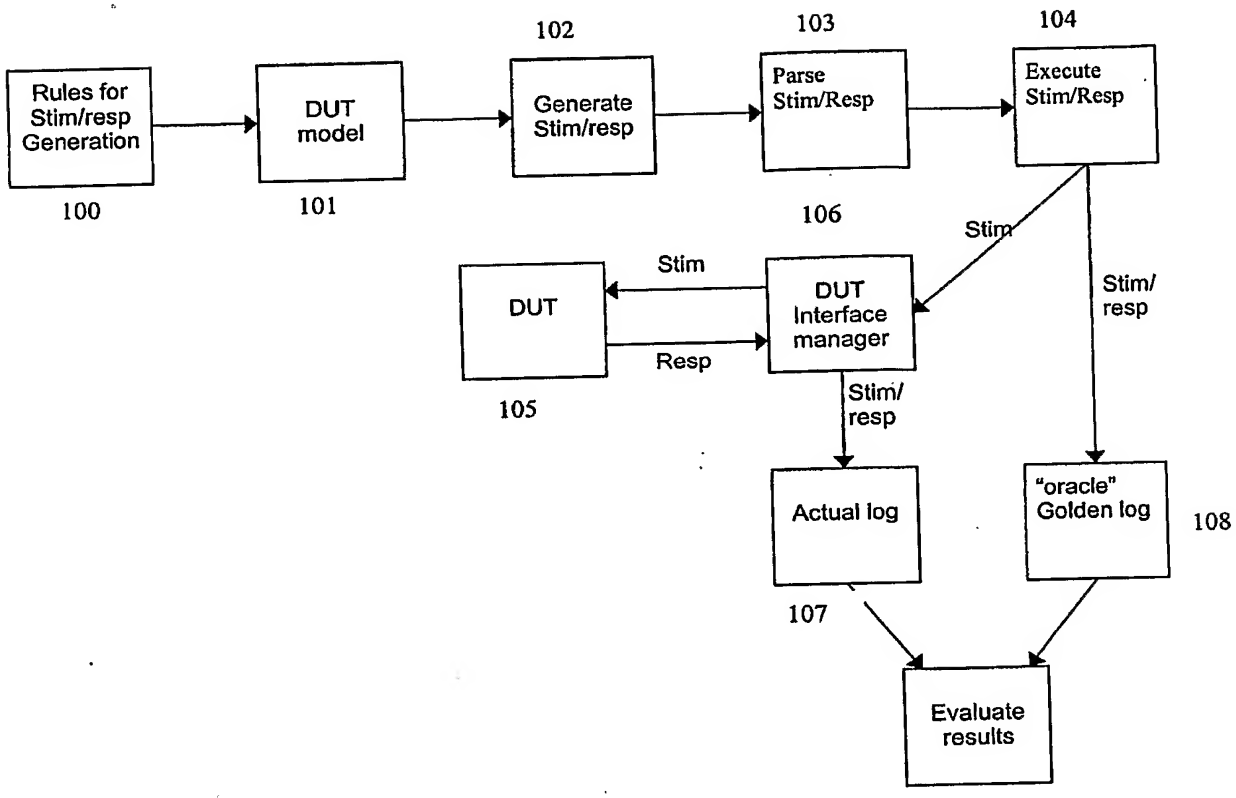
Rules for Stim/resp Generation
100
DUT model
101
102
Generate Stim/resp
103
Parse Stim/Resp
104
Execute Stim/Resp
106
Stim
Stim/ resp
DUT
105
Stim
Resp
DUT Interface manager
Stim/ resp
Actual log
107
"oracle" Golden log
108
Evaluate results

Fig 1

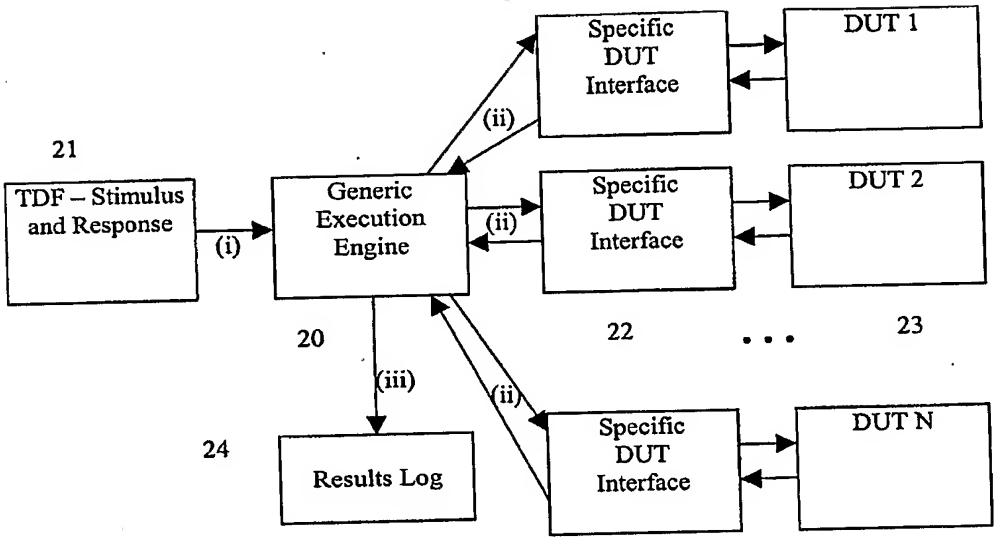


**Figure 2Generic Execution Engine Architecture**

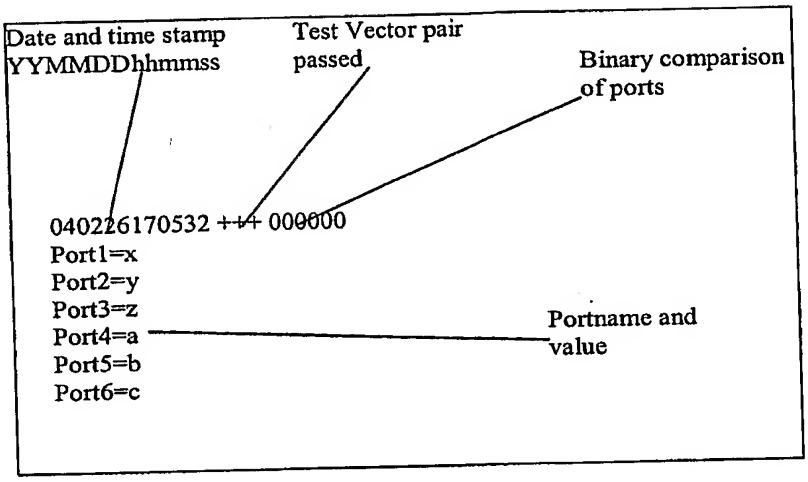
Date and time stamp
YYMMDDhhmmss
Test Vector pair
passed
Binary comparison
of ports
040226170532 ++++ 000000
Port1=x
Port2=y
Port3=z
Port4=a
Port5=b
Port6=c
Portname and
value

**Fig 3a**

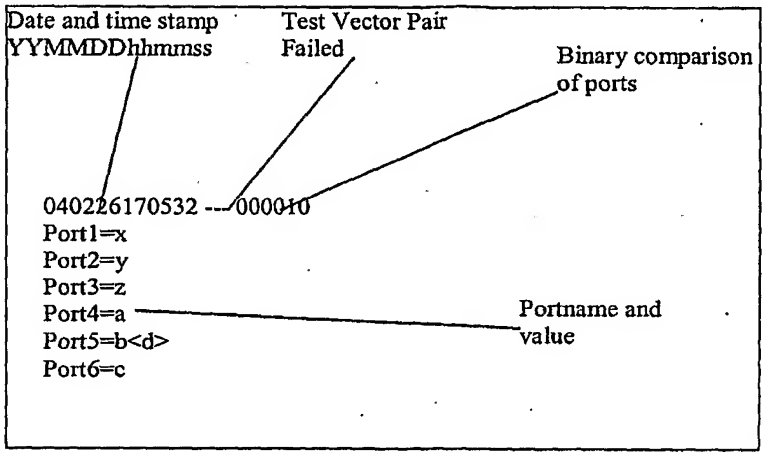
Date and time stamp
YYMMDDhhmmss
Test Vector Pair
Failed
Binary comparison
of ports
040226170532 --- 000010
Port1=x
Port2=y
Port3=z
Port4=a
Port5=b<d>
Port6=c
Portname and
value

**Fig 3b**

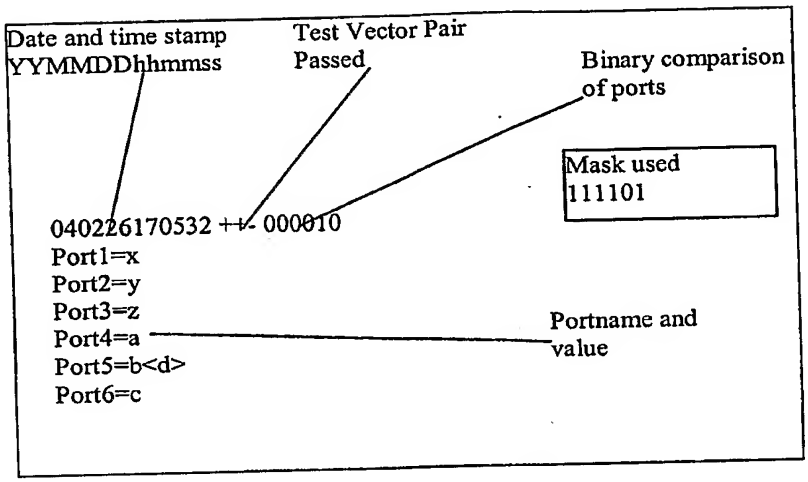
Date and time stamp
YYMMDDhhmmss
Test Vector Pair
Passed
Binary comparison
of ports
Mask used
111101
040226170532 +/- 000010
Port1=x
Port2=y
Port3=z
Port4=a
Port5=b<d>
Port6=c
Portname and
value

**Fig 3c**